Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 252 (2009) 103–119
www.elsevier.com/locate/entcs

Improvement of a Result on Sequencing Elementary Cellular Automata Rules for Solving the Parity Problem

Universidade Presbiteriana Mackenzie
Pós-Graduação em Engenharia Elétrica
Rua da Consolação 896, Consolação
01302-907 São Paulo, SP - Brazil

Pedro P.B. de Oliveira[2]

Universidade Presbiteriana Mackenzie
Faculdade de Computação e Informática & Pós-Graduação em Engenharia Elétrica
Rua da Consolação 896, Consolação
01302-907 São Paulo, SP - Brazil

Abstract

The understanding of how simple predefined computations can be achieved with cellular automata, either through individual rules, or with rules organised in spatial arrangements or in temporal sequences is a key conceptual underpinning in the general notion of emergent computation. In this context, the parity problem for cellular automata is considered here, through which the number of 1-bits in a binary string should be determined to be even or odd. Although it is known that no individual rule can solve this problem, a solution has been presented in the literature that is able to solve it, for any string length, by means of the sequential chaining of elementary cellular automata rules. Drawing on that solution, here we provide others, significantly simplified from the latter, our approach being based upon analyses of the behaviour of the rules used in the original work.

Keywords: Elementary cellular automata, parity problem, sequential co bination of rules, temporal combination of rules, rule-changing cellular automata, emergent computation.

1 Introduction

Several systems in nature exhibit sophisticated global collective information processing that emerges out of an ensemble of simple, decentralised, locally connected and interacting units. Although there is still no overall understanding of how these

[1] Email: claudio.luis.martins@terra.com.br
[2] Email: pedrob@mackenzie.br

doi:10.1016/j.entcs.2009.09.017

systems carry out such a form of *emergent* computation [9], one of the simplest systems where this notion can be studied are the cellular automata [12], [10].

Cellular automata (CAs) are discrete dynamical systems where, for each iteration, the component units of a regular lattice are updated synchronously through the action of a local transition rule. Typically, these components (or cells) are identical finite automata, organised in a single pattern of connections to other neighbouring cells, quite often under periodic boundary conditions. A transition rule provides the next state of each cell, according to the current state of the cell itself and the cell states of its neighbourhood.

Many studies have been published in order to design a specific cellular automaton rule able to perform a pre-defined computational task [13] or the appropriate use of different rules that would jointly perform a given computation, either organising them in spatially distributed arrangements ([7], [11]) or in temporal combinations, where distint rules are used in sequence ([4], [8], [5], [6], [1], [2], [3]). Regardless the case, too little is still known about how to derive any of the latter schemes, so that the target global behaviour be achieved.

The *parity problem* is a well-known task in many areas of computer science, that has also been addressed in the context of cellular automata. Such a formulation refers to a two-state, one-dimensional CA rule, in periodic boundary condition, that should be able to decide whether any initial configuration has an even or odd number of 1s, by making the lattice to converge to a configuration of only 0s or only 1s, respectively. Naturally, this is a typical case of a global problem that has to be solved by pure local processing.

Although it is easy to show that no one-dimensional CA rule can be constructed with the ability to solve this problem, [6] showed how to sequence a set of elementary cellular automata (ECA) rules - i.e., those operating over three-cell, nearest neighbours - so as to solve the parity problem. And as far as we know, this remains the only solution to the problem so far.

It is interesting that another paradigmatic, benchmark problem in emergent computation for CAs - namely, the ability for the rule to work out which bit outnumbers the other in the initial configuration - has also been proved not to have a solution for an individual rule, but various solutions do exist with temporal sequences of even elementary CA rules [8]. This makes it evident the strength of temporally combining even simple rules, so to achieve global effects over the lattice that would otherwise be impossible with arbitrarily complex rules.

Drawing on the result by [6], in the present paper we provide a simplification of that solution, our approach being derived from analyses of the behaviour of the rules used therein. In the next section only the odd size lattices are considered. The subsequent two sections are then concerned with the even size lattices: firstly, the situation of a length that is not multiple of 4 (Section 3) and, secondly, the case for when the length is a multiple of 4 (Section 4). It is worth mentioning that the same division into three lattice classes is also used in [6]. Section 5 presents some concluding remarks, including a summary of the new results, so as to make it evident how they outperform the solution of [6].

2 Solution for odd-size lattices

Before going any further, it should be pointed out that the following notation is used throughout the paper to describe a temporal combination of CA rules: $R_1^{t1}\ R_2^{t2}\ R_3^{t3}\ldots$, which stands for the application of rule 1 for t1 time steps, to the initial configuration, then the application of rule 2 for t2 time steps, to the last configuration generated out of rule 1 application, and so on. The reader should be aware that our notation - where the leftmost rule is the first one to be applied and the rightmost rule is the last - is opposite to the notation used in [6]. Notice also that all rules concerned herein are elementary CA rules.

Also, in order to represent the bit strings defining the global lattice configurations of a CA, we use the standard notation for regular expressions, summarised as follows: $b_1 + b_2$ refers either to b_1- or b_2- bit; b^n is a bit string of n concatenated b-bits; b^+ is a bit string with one or more concatenated b-bits; and b^* is a bit string with zero or more concatenated b-bits.

Now, back to the section core, for lattices with odd length N, the solution given by [6] is $\sigma_1 = \left(R_{132}^{\lfloor\frac{N}{2}\rfloor} R_{222}^{\lfloor\frac{N}{2}\rfloor}\right)^{\lfloor\frac{N}{2}\rfloor} \sigma$, where σ is the input odd size bit string, that is iterated by rule 222 for $\lfloor\frac{N}{2}\rfloor$ time steps, followed by rule 132 for $\lfloor\frac{N}{2}\rfloor$ time steps, and then the same previous double sequence is repeated until reaching $\lfloor\frac{N}{2}\rfloor$ times. Thus, the total number of iterations is $2 \times \lfloor\frac{N}{2}\rfloor^2$. For $N = 9$, for example, the expression results, in our notation, the rule sequence $R_{222}^4\ R_{132}^4\ R_{222}^4\ R_{132}^4\ R_{222}^4\ R_{132}^4\ R_{222}^4\ R_{132}^4$.

2.1 Analysis of the rules involved

By analysing the effect of rule 222, one realises that it is able to turn a bit string of the form $100^{N-4}01$ into $110^{N-4}11$, when the string of 0s has the form 00^+, therefore maintaining the parity of this kind of initial configuration. There is no effect when the rule is applied to a string without consecutive 0s or only a string of 0s (0^+). Therefore, in periodic boundary condition, a lattice of the form $0^x1^10^y$, for $x + y + 1 = N$, is transformed into $0^{x-1}1^30^{y-1}$ in one iteration of the rule.

Hence, applying rule 222 for $\lfloor\frac{N}{2}\rfloor$ iterations, no consecutive 0s are preserved, provided that at least one 1 is originally present in the lattice.

Similarly, rule 132 is able to turn a bit string of the form $011^{N-4}10$ into $001^{N-4}00$, when the string of 1s has the form 11^+, also keeping the parity of the initial configuration. And there is no effect when the rule is applied to a string without consecutive 1s or only a string of 1s (1^+). So, in periodic boundary condition, a lattice of the form $1^x0^11^y$, with $x + y + 1 = N$, is transformed into $1^{x-1}0^31^{y-1}$ in one iteration of the rule.

Therefore, applying rule 132 for $\lfloor\frac{N}{2}\rfloor$ iterations, no consecutive 1s remain in the lattice, provided at least one 0 is present in it.

2.2 Simplifying solution σ_1

Bearing in mind that all odd size lattices necessarily display at least two consecutive 1s or 0s, let us consider the worst case of when one bit is outnumbered by the other by a single unit; for lattice size $N = 15$, this corresponds, for instance, to 010101011010101.

The application of rule 222 to the latter, one or more times has no effect, since it has no adjacent 0s. Then, by applying rule 132, both 1s of the substring of running 1s are replaced by 0s, without changing the parity of the initial configuration. Carrying on for some more iterations, no further effect comes about, because of the lack of running 1s, that is,

$$\begin{array}{ll} t_0 & 010101\mathbf{011}0101 \quad \text{Start} \\ t_1 & 010101\mathbf{000}0101 \quad R_{132} \end{array}$$

Returning to rule 222, the two final 0s of the string of running 0s are then replaced by 1s, generating two blocks of two 1s separated by a block of two 0s. If the number of iterations is just one unit more than the number of iterations performed by the previous rule, it undoes the string of consecutive 0s entirely. In the example, two iterations turn all running 0s into running 1s, increasing the size of the 1-sequence because it is added with the two further 1s that were at the ends of the original block of 0s; in other words:

$$\begin{array}{ll} t_1 & 01010\mathbf{10001}0101 \quad R_{132} \\ t_2 & 01010\mathbf{11001}0101 \quad R_{222} \\ t_3 & 01010\mathbf{11111}0101 \quad R_{222} \end{array}$$

Now, applying again rule 132 to the last configuration above, for three iterations, at each iteration two terminal 1s of the sequence of running 1s are replaced by 0s, transforming the entire block of 1s in a bigger block of 0s, namely,

$$\begin{array}{ll} t_3 & 0101\mathbf{01111110}101 \quad R_{222} \\ t_4 & 0101\mathbf{00111100}101 \quad R_{132} \\ t_5 & 0101\mathbf{00011000}101 \quad R_{132} \\ t_6 & 0101\mathbf{00000000}101 \quad R_{132} \end{array}$$

Back to rule 222 again, for further four iterations, then rule 132 for five iterations, followed by rule 222 for six iterations, and finally, back to rule 132 for seven iterations, the initial configuration is correctly converted to the string 0^{15}, which is the solution to the example.

The solution is then $R^1_{132}\,R^2_{222}\,R^3_{132}\,R^4_{222}\,R^5_{132}\,R^6_{222}\,R^7_{132}$, but if the initial configuration had two consecutive 0s, as in 010101001010101, for instance, rule 132 would start without any effect, and the other rules, in sequence, would always *waste* one iteration, in the sense that the last iteration at each stage would bear no effect, as shown below (where iterations t_1, t_3, t_6 and t_{10} are wasted):

$$\begin{array}{ll} t_0 & 010101\mathbf{00}1010101 \quad \text{Start} \\ t_1 & 010101\mathbf{00}1010101 \quad R_{132} \\ t_2 & 01010\mathbf{1111}010101 \quad R_{222} \end{array}$$

$$
\begin{array}{lll}
t_3 & 0101\mathbf{0111}1010101 & R_{222} \\
t_4 & 0101\mathbf{0011}\mathbf{00}10101 & R_{132} \\
t_5 & 0101\mathbf{0000}\mathbf{00}10101 & R_{132} \\
t_6 & 0101\mathbf{0000}\mathbf{00}10101 & R_{132} \\
t_7 & 0101\mathbf{1000}\mathbf{01}10101 & R_{222} \\
t_8 & 0101\mathbf{1100}\mathbf{11}10101 & R_{222} \\
t_9 & 0101\mathbf{1111}\mathbf{11}10101 & R_{222} \\
t_{10} & 0101\mathbf{1111}\mathbf{11}10101 & R_{222}
\end{array}
$$

In order to account for that initial configuration, while preserving the same series of rule applications, one more rule has to be applied, for at least seven iterations (although any additional iteration beyond the seventh would be wasted).

As a consequence, the solution capable of solving the two individual configurations above is $S_1 = R^1_{132}\, R^2_{222}\, R^3_{132}\, R^4_{222}\, R^5_{132}\, R^6_{222}\, R^7_{132}\, R^7_{222}$.

Analogously, starting with rule 222, another solution also able to solve both previous initial configurations would be $S_2 = R^1_{222} R^2_{132} R^3_{222} R^4_{132} R^5_{222} R^6_{132} R^7_{222} R^7_{132}$.

Due to the periodic boundary condition, another worst-case initial configuration occurs when a single bit is surrounded by a trailing of the opposite bit, i.e., 000000010000000. The previous S_1 and S_2 solutions also solve the problem. Fig. 1 shows the space-time diagrams of the implementation of the two solutions, for this last initial configuration.

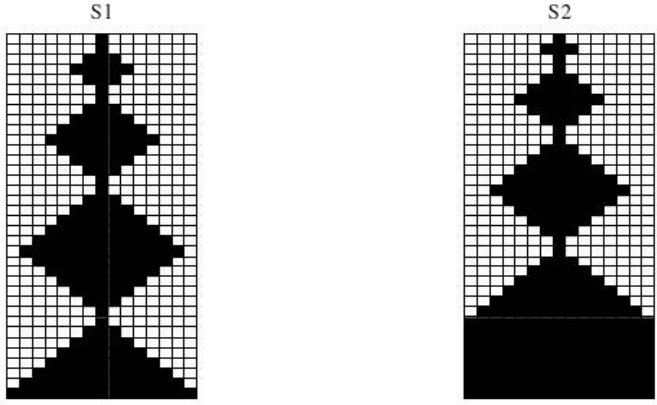

Fig. 1. Space-time diagrams resulting from the two solutions presented for the parity problem with the initial configuration 000000010000000.

For initial configurations with longer blocks of 1s or 0s, possibly in mixing quantities, all the specified iterations in S_1 and S_2 are not necessarily required, but the wasted iterations do not hinder the problem solution.

For lattice size $N = 15$, [6] showed that 98 iterations solve the problem, by switching rules 132 and 222 fourteen times. The present simplification allows starting with any of the two rules, alternating them only 8 times, totalling 35 iterations.

The number of rules needed is $\lceil \frac{N}{2} \rceil$, considering that each rule swap has to be counted as a new rule (even if a swap brings back a rule that has already been used previously in the sequence). The number of iterations required for each rule increases from 1 to $\lfloor \frac{N}{2} \rfloor$, the upper limit repeating for the final rule. The total number of iterations then becomes $\binom{(\lfloor \frac{N}{2} \rfloor^2 + 3 \times \lfloor \frac{N}{2} \rfloor)}{2}$. For example, for lattice size $N = 41$, 21 rules are needed with up to 20 iterations each, the solutions being $S_1 = R^1_{132} R^2_{222} R^3_{132} R^4_{222} \ldots R^{19}_{132} R^{20}_{222} R^{20}_{132}$ and $S_2 = R^1_{222} R^2_{132} R^3_{222} R^4_{132} \ldots R^{19}_{222} R^{20}_{132} R^{20}_{222}$, with both taking 230 iterations.

3 Solution for even-size lattices not multiple of 4

This case refers to lattice sizes $N = 2q$, for q odd, having σ is the input binary string. The solution given by [6] is $S\sigma_1 = R^{\lceil \frac{N}{2} \rceil}_{254} R_{76} \left(R^{\lfloor \frac{N}{2} \rfloor}_{132} R^{\lfloor \frac{N}{2} \rfloor}_{222} \right)^{\lfloor \frac{N}{2} \rfloor} \sigma$ or $S\sigma_1 = R^{\lceil \frac{N}{2} \rceil}_{254} R_{76} \sigma_1$ where $S = R^{\lceil \frac{N}{2} \rceil}_{254} R_{76}$ and σ_1 is the solution for odd size lattices given in Section 2. The solution means the application of the rule sequence σ_1, followed by rule 76 once, and then the application of rule 254, $\lceil \frac{N}{2} \rceil$ times. For instance, $N = 6$ results $S\sigma_1 = R^3_{222}\, R^3_{132}\, R^3_{222}\, R^3_{132}\, R^3_{222}\, R^3_{132}\, R^1_{76}\, R^3_{254}$.

3.1 Analysis of the rules involved

The relevant feature of rule 76 is that, in a single iteration, it converts a configuration of only 1s (1^+) into another of only 0s (0^+).

In contrast, rule 254 preserves configurations of only 0s, while leading all the others to converge to a configuration of only 1s. The number of iterations needed to complete this task, considering any initial configuration, is $\frac{N}{2}$.

3.2 Simplifying solution $S\sigma_1$

By applying rule sequence σ_1 to all 64 possible initial configurations with size $N = 6$, four distinct final configurations are obtained, and the same situation happens when rule sequence $S'_1 = R^1_{132}\, R^2_{222}\, R^2_{132}$ is used instead. However, when $S'_2 = R^1_{222}\, R^2_{132}\, R^2_{222}$ is used, the outcome is similar but not exactly the same as before. Table 1 summarises these outcomes.

In either of the cases, S'_1 or S'_2, the number of required rules is the same, $\frac{N}{2}$, but the number of iterations needed for each rule increases from 1 to $\frac{N}{2} - 1$, with the upper limit being the same for the last rule.

Table 1
Distinct final configurations (and their quantities) obtained from applying S_1' (or σ_1) and S_2' to all possible initial configurations of size $N = 6$.

S_1' **or** σ_1	**Quantity**	S_2'	**Quantity**
111111	16	000000	16
000000	16	111111	16
000001	30	111110	30
010101	2	101010	2

Analogously, by applying S_1 (or σ_1) or S_2 to all 1024 possible initial configurations of size $N = 10$, the same four distinct final configurations are obtained, as shown in Table 2.

Table 2
Distinct final configurations (and their quantities) obtained from applying S_1' (or σ_1) and S_2' to all possible initial configurations of size $N = 10$.

S_1' **or** σ_1	**Quantity**	S_2'	**Quantity**
1111111111	246	0000000000	246
0000000000	266	1111111111	266
0000000001	510	1111111110	510
0101010101	2	1010101010	2

Notice that the final configurations that differ from 0^N and 1^N have odd parity. Because rules 132 and 222 do not modify the initial parity, the initial configurations with even parity converge to 1^N or to 0^N, and the initial configurations with odd parity do not converge to 0^N or 1^N. Table 3 summarises the outcomes of applying S_1 (or σ_1) to the initial configurations of size $N = 18$, from where it is clear that out of the 262,144 possibilities, half of them converge to 1^N or 0^N, and half to the three other possible final configurations.

Table 3
Distinct final configurations (and their quantities) obtained from applying S_1' (or σ_1) to all possible initial configurations of size $N = 18$.

S_1' **or** σ_1	**Quantity**
111111111111111111	64,996
000000000000000000	66,076
000000000000000001	130,212
000001000001000001	858
010101010101010101	2

To complete the solution at issue, rule 76 has to be applied, by a single iteration.

Only the first configuration (1^N) is changed to 0^N, the others remaining the same. Hence, for initial configurations with even parity, the solution of the problem has already been achieved, thus now remaining to convert the other configurations to 1^N. For that, rule 254 is then used, as it preserves only the 0^N bit strings, changing all the others to 1^N. The number of iterations needed is $\lfloor \frac{N}{2} \rfloor$. The sequence below illustrates the action of rule 254 on the worst-case initial configuration, among those with odd parity, as it requires the largest number of iterations:

t_0	000000000000000001	Start
t_1	100000000000000011	R_{254}
t_2	110000000000000111	R_{254}
t_3	111000000000001111	R_{254}
t_4	111100000000011111	R_{254}
t_5	111110000000111111	R_{254}
t_6	111111000001111111	R_{254}
t_7	111111100011111111	R_{254}
t_8	111111110111111111	R_{254}
t_9	111111111111111111	R_{254}

Applying rule 76 to the configurations resulting from S_2', they are converted to 0^N. However, other configurations are changed to others that also have to be converted to 1^N by rule 254. The sequence below shows one of the configurations resulting from S_2' when it is submitted to rule 76 for one iteration:

t_0	111111111111111110	Start
t_1	100000000000000010	R_{76}

Subsequently, the application of rule 254 entails:

t_0	100000000000000010	Start
t_1	110000000000000111	R_{254}
t_2	111000000000001111	R_{254}
t_3	111100000000011111	R_{254}
t_4	111110000000111111	R_{254}
t_5	111111000001111111	R_{254}
t_6	111111100011111111	R_{254}
t_7	111111110111111111	R_{254}
t_8	111111111111111111	R_{254}

Notice that $\lfloor \frac{N}{2} \rfloor - 1$ iterations are necessary above; but for the other configurations derived from S_2, a smaller number of iterations is needed for rule 254, after rule 76 is applied by one iteration.

Therefore, in order to simplify $S\sigma_1 = R_{254}^{\lceil \frac{N}{2} \rceil} R_{76}\sigma_1$, two equivalent solutions become possible: $S_3 = S_1' R_{76}\, R_{254}^{\frac{N}{2}}$ or $S_4 = S_2' R_{76}\, R_{254}^{\frac{N}{2}-1}$.

4 Solution for even-size lattices multiple of 4

The solution given by [6] for lattice size $N = 2^m q$, with q odd and $m \geq 2$, is: $S\sigma_2 = R_{254}^{\lceil \frac{N}{2} \rceil} R_{76} \left(R_{132}^{\lfloor \frac{N}{2} \rfloor} R_{222}^{\lfloor \frac{N}{2} \rfloor} R_{184} R_{252} \right)^{m-1} \left(R_{132}^{\lfloor \frac{N}{2} \rfloor} R_{222}^{\lfloor \frac{N}{2} \rfloor} \right)^{\lfloor \frac{N}{2} \rfloor} \sigma$ or, equivalently, $S\sigma_2 = ST^{m-1}\sigma_1$, where σ is the input binary string, $S = R_{254}^{\lceil \frac{N}{2} \rceil} R_{76}$ and $T = R_{132}^{\lfloor \frac{N}{2} \rfloor} R_{222}^{\lfloor \frac{N}{2} \rfloor} R_{184} R_{252}$. This means the application of rule sequence σ_1, followed by the other rule sequence T for $m-1$ times, then rule 76 only once, and finally, rule 254 for $\lceil \frac{N}{2} \rceil$ times. For instance, for $N = 8, S\sigma_2 = R_{222}^4 R_{132}^4 R_{222}^4 R_{132}^4 R_{222}^4 R_{132}^4 R_{222}^4 R_{132}^4 R_{252}^1 R_{184}^1 R_{222}^4 R_{132}^4 R_{252}^1 R_{184}^1 R_{222}^4 R_{132}^4 R_{76}^1 R_{254}^4$.

4.1 Analysis of the rules involved

Rule 184 moves any 1-bit to the right in the lattice, if its right-hand neighbouring site is 0. Its dynamically equivalent rule, 226, does the same, but moving 1s leftwards, provided the site at the left is 0. Both arrange the lattice with alternating 0s and 1s, so that only the most frequent bit in the initial configuration can be found in consecutive sites. For lattice size N, after $\lfloor \frac{N}{2} \rfloor - 1$ iterations both rules transform any initial configuration to a bit string of the form $((01)^*1^+)^+$ if there is a predominance of 1s, to $((10)^*0^+)^+$ if there is a predominance of 0s, or to $(01)^{\frac{N}{2}}$ if there are as many 1s as 0s. Regardless the case, the same original density and parity of the initial configuration are preserved, since rule 184 is conservative.

The worst-case initial configuration is depicted in Fig. 2, as it has almost the same number of 0s and 1s, with the cells grouped in two homogeneous blocks. The lattice has 35 cells, initialised with 18 1s and 17 0s, upon which rule 184 is applied for 20 iterations.

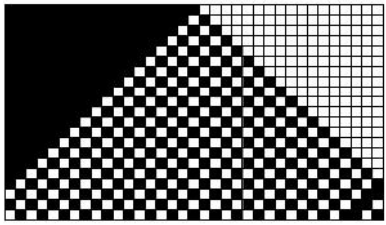

Fig. 2. Space-time diagram of the application of rule 184 on lattice size $N = 35$, with homogeneous blocks of 18 1s and 17 0s.

Furthermore, rule 252 is an auxiliary rule, that transforms a 0-bit to the right of a 1-bit in another 1.

4.2 Simplifying solution $S\sigma_2$

Similarly to the previous solution, applying rule sequence σ_1 to all 16 possible size $N = 4$ initial configurations, 4 possible final configurations are obtained, exactly the same outcome as for when $S'_2 = R^1_{222}\, R^1_{132}$ is used instead; however, applying $S'_1 = R^1_{132}\, R^1_{222}$ yields a similar result, even though not exactly the same as before. Table 4 summarises the outcomes.

Table 4
Distinct final configurations (and their quantities) obtained from applying S'_2 (or σ_1) and S'_1 to all possible initial configurations of size $N = 4$.

S'_2 **or** σ_1	**Quantity**		S'_1	**Quantity**
1111	5		0000	5
0000	1		1111	1
0001	8		1110	8
0101	2		1010	2

Notice that regardless of applying S'_1 or S'_2, only one final configuration has odd parity and three others have even parity. Here, not only the configurations 0^N or 1^N are originated from initial configurations with even parity.

Similarly, by applying S'_2 (or σ_1) or S'_1 to all 256 possible initial configurations of size $N = 8$, 5 distinct final configurations are obtained, as shown in Table 5. Notice that only one of the resulting configurations has odd parity.

Table 5
Distinct final configurations (and their quantities) obtained from applying S'_2 (or σ_1) and S'_1 to all possible initial configurations of size $N = 8$.

S'_2 **or** σ_1	**Quantity**		S'_1	**Quantity**
11111111	37		00000000	37
00000000	49		11111111	49
00000001	128		11111110	128
00010001	40		11101110	40
01010101	2		10101010	2

Now, by applying S'_2 (or σ_1) or S'_1 to all 4096 possible initial configurations of size $N = 12$, 6 distinct final configurations are obtained, as shown in Table 6. Notice that now two resulting configurations have odd parity.

Table 6
Distinct final configurations (and their quantities) obtained from applying S_2' (or σ_1) and S_1' to all possible initial configurations of size $N = 12$.

S_2' **or** σ_1	**Quantity**	S_1'	**Quantity**
111111111111	980	000000000000	980
000000000000	556	111111111111	556
000000000001	2016	111111111110	2016
000001000001	510	111110111110	510
000100010001	32	111011101110	32
010101010101	2	101010101010	2

If the same analysis above is further performed for higher values of N, it turns out that the final configurations with even parity that differ from $1^N, 0^N, (01)^{\frac{N}{2}}$ and $(10)^{\frac{N}{2}}$, and those with odd parity that differ from $0^{N-1}1$ and $1^{N-1}0$, have odd parity if the number of 0^x1 blocks is also odd, and, of course, have even parity if the number of 0^x1 blocks is also even.

After the application of S_2', all final configurations (besides 1^N and 0^N) become $(0^x1)^y$, where x is an odd number, $N = y^*(x+1)$, and $x+1$ is a factor of N. Since N is a multiple of 4, the factors 2 and $\frac{N}{2}$ allow for $x = 1$, the configuration $(01)^{\frac{N}{2}}$, and for $x = \frac{N}{2} - 1$, a configuration of the form $(0^x1)^2$. In both cases the parity is even. For $x = N - 1$, there is always a configuration of the form 0^x1, whose parity is odd.

Finally, before applying rules 76 and 254, the even parity configurations, and only these, must be converted to 1^N or 0^N. On the other hand, the odd parity configurations should be converted into any bit string, as long as they are different from 1^N or 0^N. In order to attain the latter, the following expression can be used: $T = R_{132}^{\lfloor \frac{N}{2} \rfloor} R_{222}^{\lfloor \frac{N}{2} \rfloor} R_{184} R_{252}$.

4.2.1 Analysis of the number of times $(m-1)$ expression T is subjected to

Bearing in mind the characterisation of x in the previous subsection, let us consider $N = 12$, whose factors are 1, 2, 3, 4, 6 and 12; since $x+1$ is a factor of 12 and x is odd, then x can take on the possible values 1, 3, 5 and 11.

Now, after the application of S_2', 6 possible final configurations result, namely, $(0^11)^6, (0^31)^3, (0^51)^2, (0^{11}1)^1, 0^{12}$ and 1^{12}, where only $(0^31)^3$ and $(0^{11}1)^1$ have odd parity. Expression T converts an expression of the form $(0^x1)^y$ to the form $(0^{\binom{(x-1)}{2}}1)^{2y}$, since the final configuration is one of those mentioned here, as shown below:

t_0	000000000001	Start
t_1	100000000001	R_{252}
t_2	010000000001	R_{184}

t_3	011000000011	R_{222}
t_4	011100000111	R_{222}
t_5	011110001111	R_{222}
t_6	011111011111	R_{222}
t_7	011111011111	R_{222}
t_8	011111011111	R_{222}
t_9	001110001110	R_{132}
t_{10}	000100000100	R_{132}
t_{11}	000100000100	R_{132}
t_{12}	000100000100	R_{132}
t_{13}	000100000100	R_{132}
t_{14}	000100000100	R_{132}

Notice that iterations $t_7, t_8, t_{11}, t_{12}, t_{13}$ and t_{14} are being wasted and could be eliminated.

After the application of T, the bit string $(0^{11}1)^1$ turns into $(0^51)^2, (0^31)^3$ turns into $(0^11)^6, (0^51)^2$ becomes 0^{12}, and $(0^11)^6$ is transformed into 1^{12}.

Thus, all initial configurations with even parity would converge to 0^N or 1^N, and all configurations with odd parity to a different bit string, just as in the case of lattice sizes not multiple of 4. Therefore, the application of rules 76 and 254 could conclude the problem solution.

For lattice size $N = 8$, the possible final configurations due to S_2' are $(0^11)^4, (0^31)^2, (0^71)^1, 0^8$ and 1^8. Then, a single application of T would entail, respectively, $1^8, (0^11)^4, (0^31)^2, 0^8$ and 1^8; this means that there is still an original configuration with even parity (the one converted by S_2' to $(0^31)^2$ and then converted by T into $(0^11)^4$) that should be converted to 0^8 or 1^8, before the final application of rules 76 and 254. Applying T once again would lead, respectively, to the bit strings $1^8, 1^8, (0^11)^4, 0^8$ and 1^8, meaning that all the original configurations with even parity would have been converted to 0^8 or 1^8, and the odd ones to a different bit string. Finally, if T were applied yet again, the original configurations with odd parity would have converted to 1^8. Therefore, the number of times T is applied has to be exact. This value, $m - 1$, depends on the number of factors 2 in the factorisation of N; for example, $m = 3$ for $N = 8 = 2 \times 2 \times 2$, and $m = 2$ for $N = 12 = 2 \times 2 \times 3$. So, for $N = 2mq$, with q odd and $m \geq 2$, it is necessary to apply T for $m - 1$ times.

4.2.2 Reduction in the number of iterations of rules 222 and 132 in expression T

As mentioned above, some iterations are wasted when applying rules 222 and 132. Naturally, the minimum numbers of applications required for each rule depend on N and m; but these quantities effectively decrease after each application of T, for $m > 2$; this is exemplified in Fig. 3, for $N = 48$, where 3 applications of T are necessary, since $m = 4$. The first application of T requires 22 iterations of rule 222 and 11 iterations of rule 132, the second requires 10 and 5 iterations, respectively,

and the third, 4 and 2 iterations.

The number of rule 132 iterations is always half the number required for rule 222. Notice that the first application of T requires $\frac{N}{2} - 2$ iterations for rule 222, the second requires $\frac{N}{4} - 2$, and the third $\frac{N}{8} - 2$ iterations.

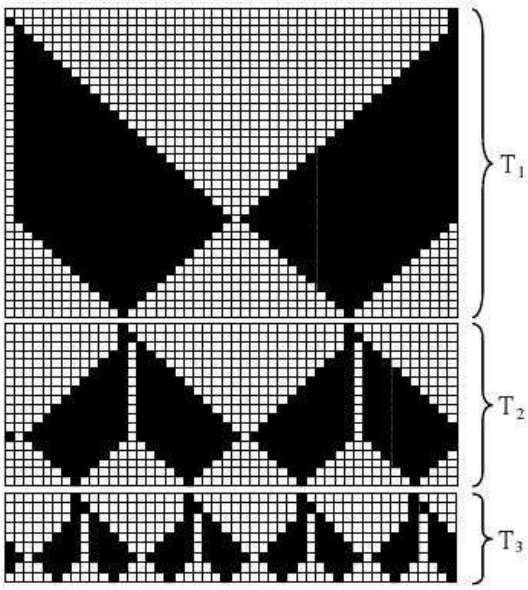

Fig. 3. Space-time diagram of the improved application of rule sequence T on a lattice size $N = 48$, starting with 47 0s and a single 1. The three stages refer to the improved versions T_1, T_2 and T_3 respectively.

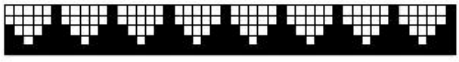

Fig. 4. Space-time diagram of the application of rule 76, for one iteration, and of rule 254, for 3 iterations.

Figure 3 shows the application of T with the numbers of iterations required for rules 222 and 132, reduced to the minimum, on a lattice size $N = 48$, from the initial configuration composed of 47 0s and a single 1. The figure depicts the three subsequent applications of T, as mentioned above.

In the first application of T, it is optimised to $T_1 = R^1_{252}\, R^1_{184}\, R^{22}_{222}\, R^{11}_{132}$ (from the original $T = R^1_{252}\, R^1_{184}\, R^{24}_{222}\, R^{24}_{132}$). One should notice that the number of 0^x1 blocks doubles, from the initial configuration $(0^{47}1)^1$ to the final configuration $(0^{23}1)^2$, as mentioned earlier. The subsequent application $T_2 = R^1_{252}\, R^1_{184}\, R^{10}_{222}\, R^5_{132}$ is then displayed in Fig. 3, making it evident once again the doubling number of 0^x1 blocks (from $(0^{23}1)^2$, at the beginning of stage 2, to $(0^{11}1)^4$, the last configuration in the same stage). Finally, the effect of applying $T_3 = R^1_{252}\, R^1_{184}\, R^4_{222}\, R^2_{132}$ is shown in the figure, where the number of 0^x1 blocks doubles once again, from $(0^{11}1)^4$, the initial configuration of the third stage, to $(0^5 1)^8$, the last one in the sequence.

So, not only T_1 reflects an improvement over T, but also T_2 and T_3, T_2 because the most laborious block $(0^{47}1)^1$ had been previously converted to $(0^{23}1)^2$ by T_1,

and T_3 because the block $(0^{23}1)^2$ had been already transformed into $(0^{11}1)^4$ by T_2.

4.2.3 Reduction in the iterations of rule 254

After the application of T, for $m-1$ times, optimised or not, it is still required to finalise the process with rules 76 and 254.

At this stage, all even parity initial configurations have been properly converted to 0^N or 1^N, and all odd parity initial configurations to $(0^x1)^Y$, where $(x+1)Y = N$. Hence, all that is required is to apply rule 76 by a single iteration, in order to conclude the process for even parity initial configurations, and to apply rule 254, for a few iterations, in order to convert the other configurations to 1^N, thus ending the case also for the odd parity initial configurations.

Now, considering the worst-case configuration $0^{N-1}1$, after the application of S_2' (or σ_2), the $m-1$ applications of T yield a configuration of the form $\left(0^{\frac{N}{2^{m-1}}-1}1\right)^{2^{m-1}}$. In the example above, with $N = 48$ and $m = 4$, 3 optimised expressions of T were applied, that resulted in $(0^51)^8$. Applying rule 76 to the latter, nothing happens; but, at each iteration of rule 254, the cells surrounding those with 1-state also turn to 1, thus converting the entire lattice to 1^N after 3 iterations. Fig. 4 illustrates this process, which represents the final step in solving the parity problem for the example configuration discussed in this section (notice that the initial configuration in Fig. 4 is the last one from Fig. 3).

As the number of 0s in each block is $\frac{N}{2^{(m-1)}} - 1$, and at each iteration of rule 254 two 0s are converted into two 1s, it turns out that $\frac{N}{2^m}$ iterations of rule 254 are necessary in order to transform all the 0s in the original bit string.

4.2.4 Equivalent rules to replace rules 184 and 252

So far, the solution we derived for even size lattices, multiple of 4 is

$$S_5 = S_2' \prod_{i=1}^{m-1} \left(R_{252} R_{184} R_{222}^{\frac{N}{2^i}-2} R_{132}^{\frac{N}{2^{i+1}}-1} \right) R_{76} R_{254}^{\frac{N}{2^m}} \tag{1}$$

Notice that this solution can also be used for even size lattices, but only those that are not multiple of 4, since $m = 1$ in those cases; therefore, all this leads to the solutions summarised below:

$$S_5 = S_2' R_{76} R_{254}^{\frac{N}{2}} \equiv S_4 = S_2' R_{76} R_{254}^{\frac{N}{2}-1} \equiv S_3 = S_1' R_{76} R_{254}^{\frac{N}{2}} \tag{2}$$

The rule sequence T can have its rules 184 and 252 replaced by their dynamically equivalent rules. Accordingly, rule 226 can directly replace its equivalent 184 in the expression above. However, rule 252, that has 3 equivalent rules (238, 192 and 136) can be replaced by rule 238 in the expressions because T ends with rule 132, and can also be replaced by rules 192 or 136, provided that T would be expressed in the different but equivalent way, where rules 222 and 132 would be reversed, namely, $T' = R_{222}^{\lfloor \frac{N}{2} \rfloor} R_{132}^{\lfloor \frac{N}{2} \rfloor} R_{184} R_{192}$.

5 Concluding remarks

Here we tackled the problem of devising a solution to the parity problem in cellular automata, which is a paradigmatic example of a global computation that has to rely entirely on local coordination of action. Although this and other similar problems are known not to be solvable by any individual CA rule, quite surprisingly solutions are possible by sequencing the extremely simple ECA rules, i.e., temporally combining ECA rule applications over time.

Although various temporal combinations for the density problem are known [8], here we set out to simplify an existing temporal combination of ECA rules to solve the parity problem, apparently the only one still known, namely, the construction given by [6]. Our successful approach - that led to more and simpler solutions - was drawn from analyses of the behaviour of the ECA rules present in the original solution, and tested in extensive computational experiments.

The methodology employed was fundamentally based on phenomenological analyses of the rules involved and their global effect. These took the form, for instance, of analysing worst-case initial configurations, whose correct processing would then determine the corresponding rule sequence upper-bounds. This is naturally a quite reasonable rationale, that turned out to be essential in determining the bottlenecks present in [6], that their purely formal approach could not unveil. Also, the idea of analysing the effect of rule substitutions in a solution for others in the same class of dynamical equivalence was key to expanding the set of equivalent solutions presented here, an idea that was drawn from our previous work on the subject [8]. Therefore, the route we took seem generalisable to other computational tasks. Nevertheless, we believe that it is very likely amenable to a more formal treatment, that would render our argument more compact.

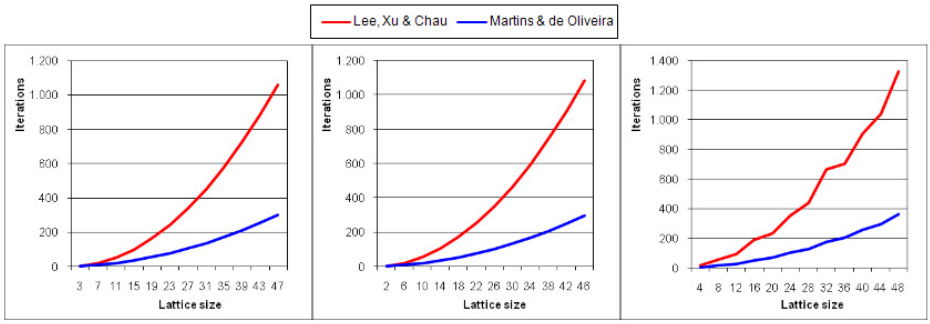

Fig. 5. Comparison between the approaches in [6] and their improved solutions, presented here, for various lattice sizes: odd size at the left, even size not multiple of 4 at the centre, and even size multiple of 4 at the right.

Table 7
The original and the improved solutions, for odd size lattices, and their corresponding number of required iterations (τ).

Original Solution	**Improved, Equivalent Solutions**
$\sigma_1 = \left(R_{132}^{\lfloor \frac{N}{2} \rfloor} R_{222}^{\lfloor \frac{N}{2} \rfloor}\right)^{\lfloor \frac{N}{2} \rfloor} \sigma$	$S_1 = R_{132}^1 R_{222}^2 R_{132}^3 \dots R_{132}^{\lfloor \frac{N}{2} \rfloor - 1} R_{222}^{\lfloor \frac{N}{2} \rfloor} R_{132}^{\lfloor \frac{N}{2} \rfloor}$
	$S_2 = R_{222}^1 R_{132}^2 R_{222}^3 \dots R_{222}^{\lfloor \frac{N}{2} \rfloor - 1} R_{132}^{\lfloor \frac{N}{2} \rfloor} R_{222}^{\lfloor \frac{N}{2} \rfloor}$
$\tau = \frac{(N-1)^2}{2}$	$\tau = \frac{(N-1)^2+6(N-1)}{8}$

Table 8
The original and the improved solutions, for even size lattices $N = 2q$ (q odd), N non multiple of 4, and their corresponding number of required iterations (τ).

Original Solution	**Improved, Equivalent Solutions**
$S\sigma_1 = R_{254}^{\lfloor \frac{N}{2} \rfloor} R_{76} \sigma_1$	$S_3 = S'_1 R_{76} R_{254}^{\frac{N}{2}}$
	$S_4 = S'_2 R_{76} R_{254}^{\frac{N}{2}-1}$
	where $S'_1 = R_{132}^1 R_{222}^2 R_{132}^3 \dots R_{132}^{\frac{N}{2}-2} R_{222}^{\frac{N}{2}-1} R_{132}^{\frac{N}{2}-1}$
	and $S'_2 = R_{222}^1 R_{132}^2 R_{222}^3 \dots R_{222}^{\frac{N}{2}-2} R_{132}^{\frac{N}{2}-1} R_{222}^{\frac{N}{2}-1}$
	$\tau = \frac{N^2+6N}{8}$ for S_3
$\tau = \frac{N^2+N+2}{2}$	$\tau = \frac{N^2+6N-8}{8}$ for S_4

Table 9
The original and the improved solutions, for even size lattices $N = 2mq$ (q odd and $m > 1$), N multiple of 4, and their corresponding number of required iterations (τ).

Original Solution	**Improved, Equivalent Solutions**
$ST^{m-1}\sigma_1 = R_{254}^{\lfloor \frac{N}{2} \rfloor} R_{76} T^{m-1} \sigma_1$, where	$S_5 = S'_2 \prod_{i=1}^{m-1} \left(R_{252} R_{184} R_{2^i}^{\frac{N}{2^i}-2} R_{132}^{2^{i+1}N-1}\right) R_{76} R_{2^m}^{\frac{N}{2^m}}$
$T = R_{132}^{\lfloor \frac{N}{2} \rfloor} R_{222}^{\lfloor \frac{N}{2} \rfloor} R_{184} R_{252}$	$S_6 = S'_2 \prod_{i=1}^{m-1} \left(R_{252} R_{226} R_{2^i}^{\frac{N}{2^i}-2} R_{132}^{2^{i+1}N-1}\right) R_{76} R_{2^m}^{\frac{N}{2^m}}$
	$S_7 = S'_2 \prod_{i=1}^{m-1} \left(R_{238} R_{226} R_{2^i}^{\frac{N}{2^i}-2} R_{132}^{2^{i+1}N-1}\right) R_{76} R_{2^m}^{\frac{N}{2^m}}$
	$S_8 = S'_2 \prod_{i=1}^{m-1} \left(R_{238} R_{184} R_{2^i}^{\frac{N}{2^i}-2} R_{132}^{2^{i+1}N-1}\right) R_{76} R_{2^m}^{\frac{N}{2^m}}$
	$S_9 = S'_2 \prod_{i=1}^{m-1} \left(R_{192} R_{184} R_{132}^{\frac{N}{2^i}-2} R_{222}^{2^{i+1}N-1}\right) R_{76} R_{2^m}^{\frac{N}{2^m}}$
	$S_{10} = S'_2 \prod_{i=1}^{m-1} \left(R_{192} R_{226} R_{132}^{\frac{N}{2^i}-2} R_{222}^{2^{i+1}N-1}\right) R_{76} R_{2^m}^{\frac{N}{2^m}}$
	$S_{11} = S'_2 \prod_{i=1}^{m-1} \left(R_{136} R_{226} R_{132}^{\frac{N}{2^i}-2} R_{222}^{2^{i+1}N-1}\right) R_{76} R_{2^m}^{\frac{N}{2^m}}$
	$S_{12} = S'_2 \prod_{i=1}^{m-1} \left(R_{136} R_{184} R_{132}^{\frac{N}{2^i}-2} R_{222}^{2^{i+1}N-1}\right) R_{76} R_{2^m}^{\frac{N}{2^m}}$
$\tau = \frac{N^2-N-2}{2} + m(N+2)$	$\tau = \frac{N^2+14N}{8} - \frac{N}{2^{m-1}} - m + 1$

Tables 7, 8 and 9 summarise the results discussed in the paper, providing all new solutions that had been derived, as well as their corresponding efforts, reflected in the number of iterations (τ) required to reach the solutions. For the sake of comparison, the same tables also present the solutions by [6] and their efforts and, additionally, Fig. 5 depicts a clearer comparison of the efforts of both approaches.

Even though it is tempting to state that the solutions we built are optimised versions of their original counterparts, there may still be some room for further improvement. For instance, a possible improvement one might consider would be a way to make the number of applications of rule sequence T more flexible; after all, both in [6] and presently a precise number of applications is required. More than an improvement in efficiency of the corresponding solutions that such a flexibilisation might entail, it would bring about a cleaner conceptual context for the related solutions.

All in all, it is quite interesting that the ECA rule space, though small and composed of extremely simple units, has continuously reaffirmed itself as a rich conceptual space, with a lot yet to explore.

Acknowledgement

We thank research grants awarded to PPBO, by FAPESP - Fundação de Amparo à Pesquisa do Estado de São Paulo (Proc. 2005/04696-3), MackPesquisa - Fundo Mackenzie de Pesquisa (Edital 2007), and Wolfram Research (Mathematica Academic Grant No. 1149). We thank Rodrigo Freitas for LaTex-ing our original manuscript.

References

[1] H.F. Chau, L.W. Siu and K.K. Yan. "One dimensional n-ary den- sity classification using two cellular automaton rules". *International Journal of Modern Physics C*, 10(5):883-889, 1999.

[2] H.F. Chau, K.K. Yan, K.Y. Wan and L.W. Siu. "Classifying rational densities using two one-dimensional cellular automata". *Physical Review E*, 57(2):1367-1369, 1998.

[3] H. Fukś. "Solution of the density classification problem with two cellular automata rules". *Physics Review E*, 55:2081R-2084R, 1997.

[4] H. Kanoh and S. Sato. "Improved evolutionary design for rule-changing cellular automata based on the difficulty of problems", *IEEE International Conference on Systems, Man and Cybernetics*, pp.1243-1248, 2007.

[5] H. Kanoh and Y. Wu. "Evolutionary design of rule changing cellular automata". *Lecture Notes in Computer Science*, 2773:258-264, Springer-Verlag: Berlin, 2003.

[6] K.M. Lee, H. Xu and H.F. Chau. "Parity problem with a cellular automaton solution". *Physical Review E*, 64:026702/1-026702/4, 2001.

[7] P. Maji and P. Pal Chaudhuri, "Non-uniform cellular automata based associative memory: Evolutionary design and basins of attraction", *Information Sciences*, 178(10):2315-2336, 2008.

[8] C.L.M. Martins and P.P.B. de Oliveira. "Evolving sequential combinations of elementary cellular automata rules". *Lecture Notes in Artificial Intelligence*, 3630:461-470, Springer-Verlag: Berlin, 2005.

[9] M. Mitchell. "Computation in cellular automata: A selected review". In: T. Gram, S. Bornholt, M. Gro, M. Mitchell and T. Pellizzari. *Non-Standard Computation*, Wiley-VCH: Weinheim, Germany, p. 95-140, 1998.

[10] P. Sarkar. "A brief history of cellular automata". *ACM Computing Surveys*, 32(1):80-107, 2000.

[11] M. Sipper. *Evolution of Parallel Cellular Machines: The Cellular Programming Approach*. Springer-Verlag, Heidelberg, 1997.

[12] S. Wolfram. "A New Kind of Science", Wolfram Media, 2002.

[13] D. Wolz and P.P.B. de Oliveira. "Very effective evolutionary techniques for searching cellular automata rule spaces". *Journal of Cellular Automata*, 3(4):289-312, 2008.